Product-Driven Generation of Action Sequences for Adaptable Manufacturing Systems

Nadine Keddis Gerd Kainz Alois Zoitl

fortiss GmbH
Guerickestr. 25
80805 Munich, Germany
{*keddis, kainz, zoitl*}*@fortiss.org*

Abstract: In times of fast changing markets and short product life-cycles manufacturing systems have to be adaptable and able to support a variety in products and product volumes. Production has to be product-driven and switching between different products should be possible with little manual intervention. We suggest an action sequence generation approach that tailors the control programs of resources to product needs. The approach requires a model of the available resources with their capabilities and internal material flow, the material flow between resources as well as a product description. An action sequence can then be generated out of these models and later translated into an executable action sequence. The action sequence can be automatically downloaded and executed on a resource. The approach is evaluated on an educational production system with industrial components.

Keywords: Flexible manufacturing systems, Information technology, Machine code, Model-based control, Planning

1. INTRODUCTION

After the peak of mass production in 1955 there has been a shift towards mass customization (Iacocca Institute, 1991; Hu et al., 2008; Koren, 2010). The revolution of manufacturing is continuing with a trend towards personalized products (Koren, 2010). This revolution is driven by customer demands that vary over time. Current markets are saturated (Westkämper and Decker, 2006) with a higher supply than demand which forces suppliers to be more flexible in their processes to be economically viable. Manufacturing systems nowadays have to support heterogeneous products with low volume in order to be able to compete in highly competitive markets. Changeable manufacturing systems as described by Wiendahl et al. (2007) are necessary to cope with turbulent environments. They are characterized by being adaptable, intelligent, and versatile (Zor et al., 2010).

The role of IT in manufacturing has increased (Vogel-Heuser et al., 2009). However, current IT systems are still too inflexible (Zor et al., 2010; Sauer and Jasperneite, 2011) and require a lot of manual efforts when changes in the production are required (Zäh et al., 2010). Especially, scheduling and planning software requires a lot of manual effort to set-up (Klöpper et al., 2009). Some approaches are available that can automatically generate schedules. Nevertheless, they cannot react to dynamic changes of the scheduling problem and are only suitable for static environments (Cheeseman et al., 2005). Additionally, automatic transformation from schedule to machine code would be beneficial to complement such tool chains.

This paper proposes an approach for automatic generation of action sequences for production resources to handle current problems in the manufacturing domain. Production resources in this context refer to hardware entities that can execute production steps and are called resources throughout this paper. Resources have to provide predefined interfaces that give access to available control code implementations to execute actions (Zoitl et al., 2013). Additionally, the approach is based on models of resource capabilities and material flows. Action sequences for the production can be automatically generated using these models. The action sequence considers available material flow information to ensure that it is valid. For this, we differentiate between external and internal material flow. External material flow describes the material flow between different resources, whereas internal material flow is limited to the material flow within a resource. The first contribution of this paper is the automatic generation of action sequences that consider product requirements as well as internal and external material flow. The second contribution is an approach to automatically make action sequences executable. The benefit of this is the reduction of manual effort when switching between different products and factory setups. A factory setup refers to a set of available resources and their relations within the factory. This is demonstrated using a modular production system.

The remainder of this paper is structured as follows: Section 2 gives an overview of available work in the field of action sequence generation. In Section 3 a brief introduction of the execution of action sequences on resources is given. The required models to enable the approach are explained in Section 4. In Section 5 the generation

process is introduced in detail. Section 6 describes the experimental setup and the evaluation of the approach. Finally, Section 7 concludes the paper.

2. RELATED WORK

There has been much work in the past years to increase software quality and re-usability of control software, e.g., (Sünder et al., 2006; Eckert et al., 2012; Sorouri et al., 2012; Zoitl and Prähofer, 2012). The approaches mainly try to develop design patterns and improve the development of control software to achieve this. However, they focus on control software of a resource only and not on generating the equivalent action sequences. Additionally, they target rigid control software and focus on how to manage variability for different resources. In our case, we also consider that different products and their variants might be produced on the same resource. The resource must be able to support different action sequences and be able to work in different factory setups containing different resources. The goal is to allow the execution of different operations on the same resource depending on the desired product. Nevertheless, such modular concepts for control software are necessary to enable such an approach. They act as a starting point for our approach.

There has been much work in the field of recipes in manufacturing as well. They are similar to action sequences but more static and defined manually for each resource.

For fixed factory setups and plant structures, standards like ISA 88 were developed by the International Standardization Association (Brandl, 2006). ISA 88 separates the description of production steps from manufacturing resources. The standard provides means to coordinate different resources based on a predefined recipe for a product. Originally this was developed for the process manufacturing, but similar concepts for the discrete manufacturing exist, e.g., PackML guideline (Arens et al., 2006).

The NAMUR association gives a recommendation for recipes and defines general requirements of recipes in NE033 (NAMUR, 2003). It gives a recommendation on how to structure recipes for discontinuous processes and mainly focuses on the process industry. It is a good foundation for introducing recipe-based operation. Nevertheless, it does not propose how these recipes can be used from a planning and scheduling perspective.

In the field of agent-based planning and scheduling there has been plenty of work to develop suitable platforms and planning strategies, e.g., (Bussmann and Schild, 2001; Gabel and Riedmiller, 2008; Leitão, 2009; Alexakos et al., 2012; Lepuschitz et al., 2013). Holonic manufacturing tackles similar problems. They are used for coordination and sequencing of manufacturing resources (McFarlane and Bussmann, 2003; Lohse et al., 2005). Such approaches focus on negotiating schedules and allocating resources for the operations. The agents execute available code depending on the negotiated schedule.

Zäh et al. (2008) propose an approach to generate schedules using different capabilities. However, they do not discuss how the generated schedules can be translated into machine-readable control commands that can be used to automatically execute them on resources.

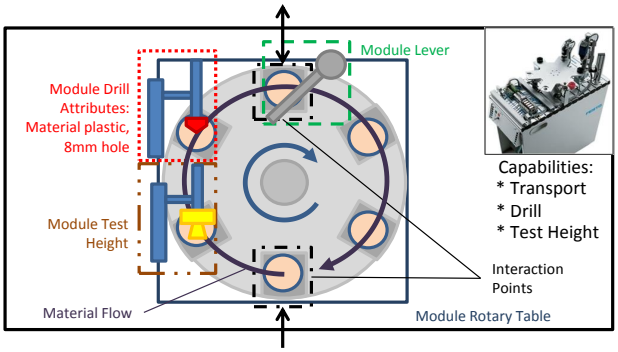

Fig. 1. Resource model for a processing station with capabilities and internal material flow information.

3. BACKGROUND

The starting point for our approach is a modular programmable logic controller (PLC) program. This is described in detail in the work of Zoitl et al. (2013). We just give a brief overview here. Every component is represented by a module comprising all functions required to control it as well as a set of interfaces to access the functions. Additionally, there is a recipe controller that is responsible for coordinating the execution of the module programs. Moreover, each program has a startup code, called program outline, which includes the basic control of a resource with its locking mechanisms. It also triggers the recipe controller that executes the currently loaded action sequence. Each command in an action sequence consists of a flag in the beginning to indicate whether this is a branch instruction or not. This is followed by a set of flags to determine the branch. Afterwards, the command is specified using an identifier for the used module, interface, and parameters. The execution of action sequences is similar to how CPUs function.

4. SYSTEM MODELING

In order to generate action sequences based on the product requirements, a factory-independent description of the product has to be provided. Additionally, a description of the factory and the available resources is necessary. As proposed in previous work (Keddis et al., 2013, 2014) the resources are modeled based on their capabilities. Capabilities of each resource express the processing steps that are supported and can be executed by this resource. We just give a brief example of the resource model, for more information see (Keddis et al., 2013). In addition to modeling capabilities, the internal material flow of the resource has to be modeled. The internal material flow describes the internal structure of a resource and how it is actually built to be able to support the processing steps. Moreover, a factory model is provided that represents how the different resources interact within the factory. Combining this information with a model for the control software of the resource can then enable an automatic generation of a product-based action sequence that can be directly executed on the resource. In the following, the different required models are presented in detail.

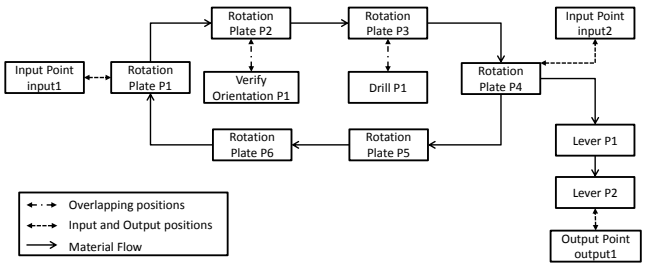

Fig. 2. Resulting internal material flow information for a processing station.

4.1 Modeling Resources

A resource model consists of all capabilities of a resource as well as its internal material flow. A resource is composed of several modules that offer a specific capability. Fig. 1 illustrates an example of such a model. The processing station is composed of a drill module, a testing module, a rotary table module, and a lever module. All modules are composed of actuators and sensors. The rotary table and the lever provide a transport capability. The drill provides a drill capability. The testing module provides a test orientation capability.

The internal material flow is used to describe how material can get from one place to another within a resource. It describes the logical flow of material. Since we are looking at adaptable manufacturing systems, this is necessary to be able to use different modules of a resource for different products. To ensure that the paths between the modules used in an action sequence exist, the internal material flow has to be modeled. Therefore, each module in the resource model additionally has at least one position. Modules that can transport material have several positions, describing all the places the material can be transported to and from. Furthermore, the links between different positions of one module can be unidirectional or bidirectional. This depends on the type of the module. A conveyor belt for example can transport in one or both directions. Positions of different modules can overlap to model that the material can be handed over from one module to the next. Overlapping positions are physically the same position but logically they indicate that material can move in both directions between modules (bidirectional flow). In the example in Fig. 1 the rotary table has 6 positions, the lever 2, and the other modules have one position each. Both the rotary table and the lever have a unidirectional material flow between their different positions. The testing module and the rotary table overlap in the second position of the rotary table. The drill and the rotary table overlap in the third position of the rotary table. Finally, the lever and the rotary table overlap in the fourth position of the rotary table. As a last step in the modeling process, the inputs and outputs of the resource have to be mapped to a position of a module. This is similar to the concept of overlapping positions. The only difference is that material can only be moved in one direction. It is either transported from the input to the module or from the module to the output. Fig. 2 shows the resulting material flow graph for the processing station.

4.2 Modeling Factories

After modeling the resource capabilities and internal material flow, instances of the resource models can be combined to model a whole factory. Additionally, external material flow information has to be added to the model to ensure that only feasible action sequences are generated. The factory model stores information about the current factory setup with available resources and their relations. The relations are represented by connections between different resources. This is the required external material flow information that is necessary to ensure that the material can later on get to the right resource. Each connection is a link between two interaction points, each belonging to a different resource. The interaction points for the processing station are illustrated in Fig. 1. The connection also has a direction that is inferred based on the types of interaction points that are linked together. Connecting an output of resource X with an input of resource Y results in a unidirectional connection from X to Y. Bidirectional connections are a result of connecting input/output points of different resources. The arrow above the interaction points in Fig. 1 indicates whether the material flow is bidirectional or unidirectional. In the latter case it also depicts the direction. The graph in the lower left box in Fig. 3 depicts a factory model with external material flow information represented by the arrows in the graph. The factory model can be automatically generated at run-time as described in our previous work (Keddis et al., 2013).

4.3 Mapping Resource Interfaces to Control Software

Executable action sequences can only be automatically generated when there is a defined interface with the corresponding control software for each module. For each required operation there has to be a mapping to the commands that have to be executed on the resource. Since we use a modular concept, the operations are specific to a module and thus, are attached to the module description. For each operation that is supported by a module, there is an executable command attached that can be downloaded to the control device of the resource in case this operation is required for the current product. For the processing station example there are interfaces related to each of the modules. The rotary table provides a *transport* interface with two positions as parameters. Whenever a transport operation using the rotary table is required, two positions are passed to the module and the corresponding control code is executed. The first position in that case describes from where to transport the material while the second position corresponds to the position where the material should go to. Similarly, the lever module provides an interface to its *transport* operation. The drill and the testing module offer an interface to their operation. This is a *drill* interface and a *test height* interface respectively. Each module implements an interface that contains the execution of the operation on this specific module. Two resources that both have two different types of e.g. a drill module will use the same interface to access the drill functions, but might have two different implementations of this interface.

4.4 Product Description

The product is described by its required production steps and their dependencies. Each of the production steps is described as a capability. The capabilities are the same as used in the resource model to enable an automatic mapping of production steps to resources. A simplified example is illustrated in Fig. 3 in the upper left box.

5. GENERATING ACTION SEQUENCES

In adaptable manufacturing systems several products can be produced using the same resources. If it is possible, the factory setup is reused for the different products and only the action sequences are updated accordingly. In some cases a change in the factory setup might be necessary. Changes include adding or removing resources, or changing the position of a resource within the factory. It is also possible to use different modules of a resource for different products. In order to automatically generate action sequences for the different products, two aspects have to be considered during the generation process: the product description and the factory setup. These can then be combined to automatically generate executable action sequences. Fig. 3 illustrates this workflow.

5.1 Automatic Action Sequence Generation

Based on the required capabilities defined in the product description and the factory setup, action sequences can be generated automatically. The factory model mentioned earlier contains all the needed information about the factory setup. It includes all available resources with their capabilities and internal material flow description as well as all the connections between different resources that model the external material flow. The first step in the automatic generation of action sequences maps each production step to all the resources providing the required capability. The mapping can be generated by a simple matching algorithm that iterates over all available resources and checks whether they can provide the capability or not. If there is no matching resource for one production step, then the product cannot be produced using the current factory setup. The algorithm stops in that case and no action sequences are generated. In case there is a match, the production step is mapped to the corresponding module that supports the capability on the resource.

The second step generates a valid action sequence considering the internal and external material flow. To find a valid action sequence we suggest using a branch-and-bound complete search with backtracking. The action sequence is generated backwards starting with the last production step of the product description. This ensures that the precedence relations described by the dependencies in the production plan are maintained because no operation is planned before its predecessors have finished. The resource mapping to the production steps introduced earlier is the starting point for the generation process. After finding suitable resources for each operation, we have to check whether there are valid internal and external material flows when these resources are used. The algorithm that generates a valid action sequence considering the external material flow is described in our previous work (Keddis et al., 2014). In this paper, we focus on integrating the internal material flow as well and generating executable code out of it.

The action sequence is generated iteratively. While generating the action sequence, we have to consider the internal material flow in addition to the external material flow in each step. Let's assume that the required operation in that step is mapped to module m_1 on resource r_1. If the previous operation was assigned to a module m_2 on the same resource (r_1), then we only have to check whether there is a valid internal material flow from m_1 to m_2. The external material flow is irrelevant in this case. If the previous operation was assigned to a module m_3 on a different resource (r_2), then both the internal and external material flows are relevant. We first check whether we can get from the input of r_1 to m_1. If r_1 has several inputs, we consider the one connected to r_2. The material can flow from the input to the module if there is a path in the internal material flow graph as illustrated in Fig. 1. The path is determined using a breadth-first-search. Since this returns the shortest path, we ensure hereby that we do not consider overlapping positions and modules and thus the path does not contain unnecessary detours through the resource. For each intermediate module a transport operation is added, so that the material can be transported from the input to m_1. We store these steps in the action sequence. The same procedure is done to determine how to get from m_3 to the output of r_2. Then we check if there is a valid external material flow from r_2 to r_1. For each intermediate resource, we determine a valid path from the input to the output of this resource and add all intermediate transport operations to the action sequence as done while checking the internal material flow. The previous steps are repeated for each operation in the product description until a valid action sequence is found. If several modules of a resource are required in consecutive steps, we first check whether we can get from the input to the first required module. Then the path between all the required modules is determined. Finally, we check how to get from the last module to the output. The search is stopped whenever there is no valid internal or external material flow and backtracking to the next branch is triggered. There is no valid action sequence when there are no backtracking branches left and no previously evaluated branch in the search tree has led to a valid action sequence. The resulting action sequence is depicted in Fig. 3 in the middle. The color of the ellipse indicates the resource that should execute the operation. The text in the ellipse indicates which operation has to be executed and which module is assigned for it.

5.2 Executable Action Sequences

After generating a valid action sequence, it is translated into an executable one. The translation is based on the interfaces that are defined for each module of the resource. Each module includes a mapping from its interfaces that are used in the generated action sequence to the corresponding control code. The planning component informs each resource which modules should be used and which commands are required and the resource defines the required executable action sequences by invoking the respective modules. For each operation in the action se-

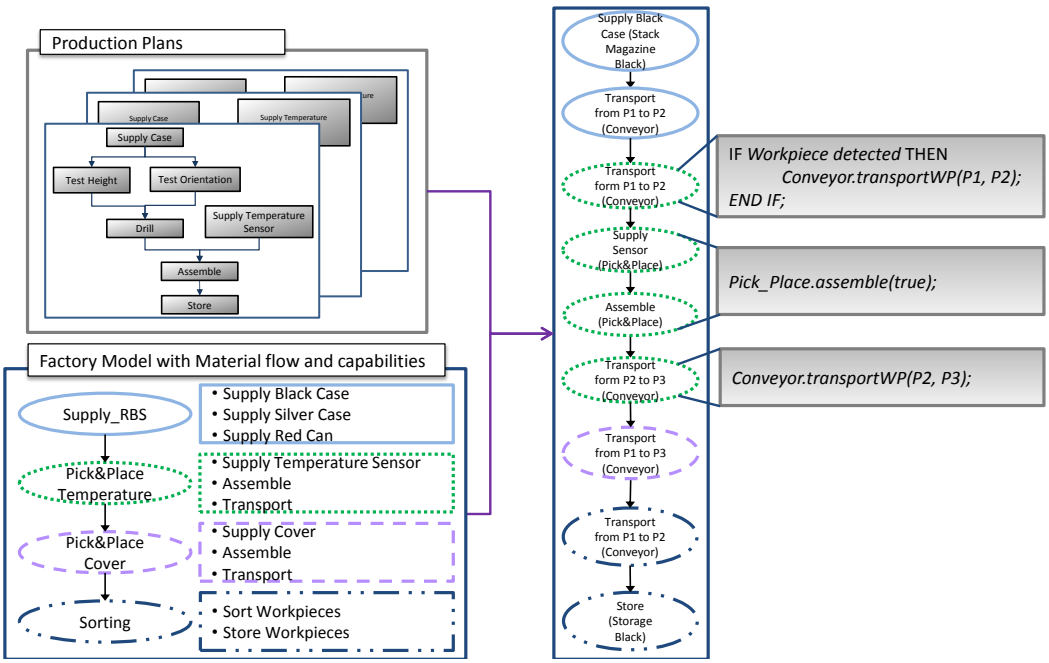

Fig. 3. The workflow for generating action sequences. A valid action sequence is generated from the product and factory descriptions. This is later translated into an executable action sequence.

quence, the executable is then composed of the required code snippets. For example if we only want to drill on the processing station, the action sequence would include a transport operation on the rotary table from the input position to the drill, a drill operation on the drill, a transport operation on the rotary table from the drill to the lever, and a transport operation from the lever to the output position. The processing station would then translate these four operations to the necessary control code. The control code includes the module that should execute the operation, the operation it should execute, and the parameters that are specific for this product. An example of the translation is depicted in Fig. 3 on the right. In general, this results in one executable action sequence that includes all the control code that is required by a resource. In case a resource is used at two different times in the action sequence, several executable action sequences can be generated for each point in time. Each executable action sequence is then automatically downloaded to the resource when the corresponding operation should start. Afterwards, the resource processes the executable action sequence and performs the operation included in it. The download process is illustrated in Fig. 4.

6. EXPERIMENTAL SETUP AND RESULTS

6.1 Experimental Setup

A simplified industrial manufacturing system is used to evaluate the approach. The setup is used for educational purposes and consists of different combinable resources from the Festo modular production system. One possible setup is shown in Fig. 4 on the right [1]. Different products can be produced within the manufacturing system: black, red, or silver thermometers, black, red, or silver hygrometers, red boxes, black, red, or silver workpieces with different characteristics. The different characteristics allow changing the production steps required for a product to demonstrate that different action sequences are generated for different products. The manufacturing systems can be composed of different Festo stations, conveyor belts, and mobile robots. Each of them provides a set of capabilities to support the production of the above mentioned products. To increase variability, production steps supported by the stations can be included or skipped in the production plan. Optional steps are testing, verifying orientation, drilling, and assembly. Stations are either controlled by a Siemens S7 300 PLC or a Festo CoDeSys PLC. Additionally, a PC is used as a supervisory control. The supervisory control calculates the valid action sequences based on the resource models and the factory model. The models are defined using the Eclipse Modeling Framework (EMF). EMF4CPP is used to generate C++ classes out of the models. Besides, a graphical user interface (GUI) runs on the supervisory control to enable user interaction. The user can specify the type of product that should be

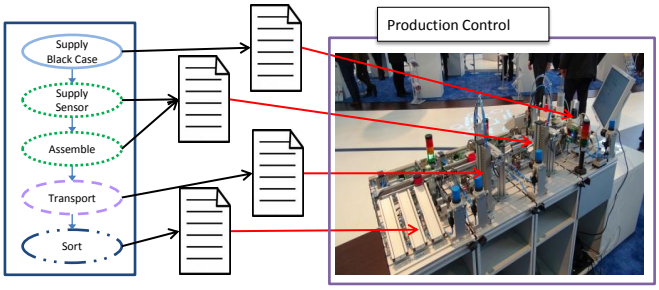

Fig. 4. A simplified action sequence is divided into an executable action sequence for each resource and downloaded to it.

[1] The experimental setup and the results can be seen in the following video: http://www.youtube.com/watch?v=Tkcv-mbhYqk.

produced in the GUI and trigger the generation of action sequences that are afterwards automatically executed. Several products can be selected at the same time. If the selected products cannot be produced on the current setup, this is displayed in the GUI by highlighting the missing capabilities. Missing links between resources can also be highlighted in the GUI.

6.2 Evaluation

For the evaluation we produce different products that require different modules of the resources to be active. For each product a valid action sequence can be found within seconds without manual configuration or programming. Product descriptions included up to ten production steps and the factory setup was composed of up to seven resources. Since the number of machines that have to be considered in each step is much smaller than the number of operations, the search space is small. Branches can be bound as soon as there is no resource that provides the capability, there is no external material flow, or there is no internal material flow. This can quickly limit the search and only a few branches of the search tree reach the last level. After generating a valid action sequence, this was successfully translated into executable action sequences. The translation was done automatically and no further manual changes were necessary to execute the action sequences. When changes in the factory setup occurred, this was automatically reflected in the generated action sequence in form of a changed number of transport operations for different modules on different resources. In order to use this approach, the resource models and the product models have to be created manually once in the beginning. The factory model can be automatically generated as described in previous work (Keddis et al., 2013). Afterwards, the models can be used for different products and factory setup. This procedure is less-error prone than traditional engineering because models can be thoroughly tested and reused later. The program outline and the recipe controller for the resource have to be programmed once manually and later the generated action sequence can be downloaded and executed automatically. The demonstration showed that the proposed approach is suitable for adaptable manufacturing systems where there are frequent switches between different products and factory setups, because it reduces manual effort, while ensuring that the production can be executed correctly. The approach still has to be evaluated on large problems.

7. CONCLUSION

In this paper we proposed an approach to automatically generate action sequences for adaptable manufacturing systems. The approach takes the requirements of the product into consideration and generates an action sequence that can be executed directly in the given manufacturing system. For this, a valid action sequence is generated based on resource and factory models, as well as mapping from interfaces to control software for each module. Thus, the action sequence generation adapts to changes in the manufacturing system in addition to changed product descriptions. It is suitable for adaptable manufacturing systems that have to deal with lots of product variants and different products. A simplified industrial setup was used to evaluate the approach. For different products and factory setups different action sequences were generated without manual effort and user interaction. The action sequences were automatically downloaded to the resources when needed and the production was triggered and executed without further intervention. The contribution in this work is an adaptable planning algorithm that adapts resource control programs depending on product requirements and factory topology.

Currently, the approach is limited to small problems. The next step in our work involves evaluating the approach on larger problems. Additionally, we intend to expand the approach to other PLC systems. Currently, the approach was tested with Siemens S7 only. Other technologies should be supported in the future. For this, the interpretation and execution of the generated recipes must be implemented on the different PLCs.

AKNOWLEDGMENT

This approach is a result of the research and development project “SpeedFactory”, which was funded by the German Federal Ministry for Economic Affairs and Energy as part of the technology programme “Autonomics for industry 4.0” and supervised by the German Aerospace Center (DLR) under Grant No. 01MA13002B.

REFERENCES

Alexakos, C., Georgoudakis, M., Kalogeras, A., and Likothanassis, S. (2012). Adaptive Manufacturing Utilizing Ontology-driven Multi-Agent Systems: Extending Pabadis’ Promise Approach. In *IEEE International Conference on Industrial Technology (ICIT)*, 42–47. IEEE.

Arens, D., Hopfgartner, T., Jensen, T., Lamping, M., Pieper, M., and D., S. (2006). *Packaging Machine Language V3.0 Mode & States Definition Document*. OMAC Motion for Packaging Working Group.

Brandl, D. (2006). *Design Patterns for Flexible Manufacturing*. ISA.

Bussmann, S. and Schild, K. (2001). An Agent-based Approach to the Control of Flexible Production Systems. In *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 2, 481–488. IEEE.

Cheeseman, M., Swann, P., Hesketh, G., and Barnes, S. (2005). Adaptive Manufacturing Scheduling: A Flexible and Configurable Agent-based Prototype. *Production Planning & Control*, 16(5), 479–487.

Eckert, K., Hadlich, T., Frank, T., Fay, A., Diedrich, C., and Vogel-Heuser, B. (2012). Design Patterns for Distributed Automation Systems with Consideration of Non-Functional Requirements. In *IEEE International Conference on Emerging Technologies & Factory Automation (ETFA)*. IEEE.

Gabel, T. and Riedmiller, M. (2008). Adaptive Reactive Job-Shop Scheduling with Reinforcement Learning Agents. *International Journal of Information Technology and Intelligent Computing*, 24(4).

Hu, S., Zhu, X., Wang, H., and Koren, Y. (2008). Product Variety and Manufacturing Complexity in As-

sembly Systems and Supply Chains. *CIRP Annals-Manufacturing Technology*, 57(1), 45–48.

Iacocca Institute (1991). 21. Century Manufacturing Enterprise Strategy: An Industry-Led View. Technical report, Iacocca Institute, Bethlehem, Pennsylvania.

Keddis, N., Kainz, G., Buckl, C., and Knoll, A. (2013). Towards Adaptable Manufacturing Systems. In *IEEE International Conference on Industrial Technology (ICIT)*, 1410–1415. IEEE.

Keddis, N., Kainz, G., and Zoitl, A. (2014). Capability-based Planning and Scheduling for Adaptable Manufacturing Systems. In *IEEE International Conference on Emerging Technologies & Factory Automation (ETFA)*. IEEE.

Klöpper, B., Sondermann-Wölke, C., Romaus, C., and Vöcking, H. (2009). Probabilistic Planning Integrated in a Multi-level Dependability Concept for Mechatronic Systems. In *IEEE Symposium on Computational Intelligence in Control and Automation (CICA)*, 104–111. IEEE.

Koren, Y. (2010). The Global Manufacturing Revolution. *John Willey & Sons, New Jersey*.

Leitão, P. (2009). Agent-based Distributed Manufacturing Control: A State-of-the-Art Survey. *Engineering Applications of Artificial Intelligence*, 22(7), 979–991.

Lepuschitz, W., Groessing, B., Merdan, M., and Schitter, G. (2013). Evaluation of a Multi-agent Approach for a Real Transportation System. In *IEEE International Conference on Industrial Technology (ICIT)*, 1273–1278. IEEE.

Lohse, N., Hirani, H., Ratchev, S., and Turitto, M. (2005). An Ontology for the Definition and Validation of Assembly Processes for Evolvable Assembly Systems. In *The 6^{th} IEEE International Symposium on Assembly and Task Planning: From Nano to Macro Assembly and Manufacturing (ISATP)*, 242–247. doi: 10.1109/ISATP.2005.1511480.

McFarlane, D.C. and Bussmann, S. (2003). Holonic Manufacturing Control: Rationales, Developments and Open Issues. In *Agent-based manufacturing*, 303–326. Springer.

NAMUR (2003). *NE 033 Requirements to be met by Systems for Recipe-Based Operations*.

Sauer, O. and Jasperneite, J. (2011). Adaptive Information Technology in Manufacturing. In *CIRP Conference on Manufacturing Systems*. Madison, WI, USA.

Sorouri, M., Patil, S., and Vyatkin, V. (2012). Distributed Control Patterns for Intelligent Mechatronic Systems. In *IEEE Conference on Industrial Informatics (INDIN)*. IEEE.

Sünder, C., Zoitl, A., and Christoph, D. (2006). Functional Structure-based Modelling of Automation Systems. *International Journal of Manufacturing Research*, 1(4), 405–420.

Vogel-Heuser, B., Kegel, G., Bender, K., and Wucherer, K. (2009). Global Information Architecture for Industrial Automation. *atp*.

Westkämper, E. and Decker, M. (2006). *Einführung in die Organisation der Produktion*. Springer.

Wiendahl, H.P., ElMaraghy, H., Nyhuis, P., Zäh, M.F., Wiendahl, H.H., Duffie, N., and Brieke, M. (2007). Changeable Manufacturing – Classification, Design and Operation. *CIRP Annals-Manufacturing Technology*, 56(2), 783–809.

Zäh, M.F., Beetz, M., Shea, K., Reinhart, G., Stursberg, O., Ostgathe, M., Lau, C., Ertelt, C., Pangercic, D., Rühr, T., et al. (2008). An Integrated Approach to Realize the Cognitive Machine Shop. In *Proceedings of the 1^{st} International Workshop on Cognition for Technical Systems*, 6–8.

Zäh, M.F., Reinhart, G., Ostgathe, M., Geiger, F., and Lau, C. (2010). A Holistic Approach for the Cognitive Control of Production Systems. *Advanced Engineering Informatics*, 24(3), 300–307.

Zoitl, A. and Prähofer, H. (2012). Guidelines and Patterns for Building Hierarchical Automation Solutions in the IEC 61499 Modeling Language. *IEEE Transactions on Industrial Informatics*, PP(99), 1. doi: 10.1109/TII.2012.2235449.

Zoitl, A., Kainz, G., and Keddis, N. (2013). Production Plan-Driven Flexible Assembly Automation Architecture. In *Industrial Applications of Holonic and Multi-Agent Systems*, 49–58. Springer.

Zor, S., Görlach, K., and Leymann, F. (2010). Using BPMN for Modeling Manufacturing Processes. In *Proceedings of 43^{rd} CIRP International Conference on Manufacturing Systems*, 515–522.